Porting Cultural Repositories to the Semantic Web

Borys Omelayenko

Vrije Universiteit Amsterdam, the Netherlands
b.omelayenko@cs.vu.nl,
WWW: http://borys.name

Abstract. In the eCulture project we ported a dozen datasets of different cultural institutions to the Semantic Web. In this paper we share our experiences: we sketch the technical data conversion problem, describe the conversion rules that were needed, and the methods that we used to align terms to vocabularies. We wrap it up with the statistics that give some insight on practical conversion cost and its success rate.

1 Problem

Let us look at the eCulture [Schreiber et al., 2006] demonstrator,[1] a prototypical Semantic Web application that won the Semantic Web challenge award at ISWC 2006. eCulture is a semantic search engine that allows simultaneously searching collections of multiple cultural heritage institutions. This is done by porting these collections to RDF[2] and linking collection instance objects via shared vocabularies, thus, building a large RDF graph. Then, during the search, this graph is traced and some subgraphs are extracted and returned as a result [Wielemaker et al., 2007]. In this paper we will focus on the problem of porting collections to RDF referring the reader to the demonstrator to see what can be done with it.

In eCulture we developed a methodology for porting cultural repositories to Semantic Web and RDF [Tordai et al., 2007]. This methodology is based on the fact that we typically can expect two kinds of data from a cultural heritage institution:

- meta-data describing cultural objects and their photos,
- local vocabularies that are used in some of these meta-data descriptions.

The institutions tend to use different formats and different data models to represent their collections and vocabularies, where databases, XML files, or even tab-separated plain text files are common choices.

Following the methodology, we first transform the schema of both meta-data and vocabularies to standardized RDF schemas, where Dublin Core[3] is used to describe the meta-data and SKOS[4] is used to represent the vocabularies. Then,

[1] e-culture.multimedian.nl
[2] http://www.w3.org/RDF/
[3] http://dublincore.org/
[4] http://www.w3.org/2004/02/skos/

we align local vocabulary terms with their counterparts from some standard vocabularies. Finally, we process the meta-data values to extract possible terms eventually used there, and align these terms need to the standard terms as well.

There exist a number of large standard vocabularies known and watched in the cultural heritage domain. These are the Getty[5] vocabularies: AAT (Art and Architecture Thesaurus of 34,000 concepts and 131,000 terms), ULAN (the Union List of Artist Names with 120,000 records and nearly 300,000 names), TGN (the Thesaurus of Geographic Names describing more than a million places). In addition, museums working in a specific niche tend to create shared vocabularies, such as the Dutch SVCN[6] ethnographic thesauri of more than 10,000 terms, or the Dutch artist list RKD holding around 400,000 artist records.

Technically, in the repository conversion task we receive the following inputs:

- Data sources describing meta-data and vocabularies. They may be described according to various data models stored as databases, XML dumps, or formatted text files.
- The target data models to which we need to convert the data (Dublin core and SKOS).
- The standard vocabularies to which we need to align the eventual terms used in the meta-data and local thesauri.

Given this input we need to develop a system that converts the source data to the target schema, and extracts and aligns possible terms to the standard terms. In the rest of this paper we present the technology that we built to construct such a system.

2 State of art

When converting something in the XML world, we need to start with XSLT,[7] the XML transformation language that allows creation of rules to translate on XML document to another. It has high-quality tool support and forms the standard solution for XML transformation. Our target format, RDF, is a specific kind of XML, and RDF documents can be easily constructed at the XML (syntactical) level with XSLT. In this way XSLT is widely used to perform syntactical conversions [Papotti and Torlone, 2004,Butler et al., 2004]. However XSLT is not suitable for semantic conversion and enrichment, or the porting problem as we face it:

- The data is often provided by cultural institutions in the form of databases or text files. XSLT requires the source data to be presented in XML.
- The terms need to be looked up in separate large vocabularies with some reasoning. XSLT does not provide any means to load them and look them up.

[5] `http://www.getty.edu/research/conducting_research/vocabularies/`
[6] `svcn.nl`
[7] `http://www.w3.org/TR/xslt`

– As we found out, nearly every dataset requires some dataset-specific code to be written and integrated. XSLT is not really meant for being integrated with custom code.
– Filtering broken images is nearly impossible to implement in XSLT.

Some approaches, such as [Sperberg-McQueen and Miller, 2004], perform automatic conversion of XML documents to RDF. However, they typically assume the target RDF model to follow the source XML model, that is really not the case in the cultural heritage conversion, as we will see later.

Practical state of art consists of custom program code written in general-purpose programming languages to convert a specific dataset. For example, manual creation of mappings is used in the cultural heritage domain as described in [Kondylakis et al., 2006]. While in [Kondylakis et al., 2006] an attempt to come up with a declarative language for conversion is made, in eCulture we quickly abandoned this path as, to solve practical conversion problems, it requires a language as expressive as a programming language.

The main goal of this work was to perform the conversions needed by the eCulture project. Accordingly, we started with programming specific converters, separating conversion rules for reuse. When programming we were constantly unifying the rules, maximizing their reuse. These reusable rules form some kind of a declarative rule language for conversion that is easy to be augmented with custom code.

3 Data models

We would illustrate our story with the sample source record, a simplified version of the description of the Rembrandts 'Night watch' from Rijksmuseum of Amsterdam.

3.1 Source models

A fragment of an XML dump produced out of the Rijksmuseum's information system is presented in Figure 1.

The key property of the datasets is that they describe objects that are separatable at the schema level, i.e. there is an XML path that wraps the description of each object. In the example from Figure 1 objects are wrapped with tag `recordList/record`. As we found them, these objects always have a tag for a unique identifier, such as tag `object.number` in our example. [8]

The other key property of the source objects is that they may include (repeating) parts. These parts would be representing other objects with their properties, that are treated as an integral part of the whole object. Similar to the wholes, the part objects are always wrapped up into a known XML tag. The parts occur

[8] In an SQL-accessible repository it is always possible to construct a query that would return one or more rows for each object, with one identifying field, when all the rows describing a single object would have the same value of the identifying field.

```
<recordList>
 <record>
  <maker>
   <maker>
    <name>Rembrandt Harmensz. van Rijn</name>
    <name>Rembrandt</name>
    <name>Rembrandt van Rijn</name>
    <birth_on>1606-07-15</birth_on>
    <birth_place>Leiden</birth_place>
   </maker>
   <maker.role>
    <term>schilder</term>
   </maker.role>
  </maker>
  <title>
   <title>Het korporaalschap ... bekend als de 'Nachtwacht'</title>
   <title.gb>The company ... known as 'The nightwatch'</title.gb>
   <title.type>display</title.type>
  </title>
  <title>
   <title>De Nachtwacht</title>
   <title.type>secondary</title.type>
  </title>
  <object.number>SK-C-5</object.number>
  <date>1642</date>
  <reproduct>
    <reproduction_reference>M-SK-C-5-00</reproduction_reference>
  </reproduct>
 </record>
</recordList>
```

Fig. 1. Sample XML description of 'The Nightwatch' by Rembrandt

quite frequent in the datasets. In our example we have several of them: multiple makers (wrapped with tag `maker`), titles, and reproductions (tag `reproduct`).

While the objects are always identifiable, the parts typically have no identifying property, e.g. tag `title` representing part-objects without any identification. [9]

3.2 Target models

For the meta-data we use Dublin Core, the leading ontology for meta-data descriptions of resources. It specifies resources with properties like `title`, `creator`,

[9] In SQL results the parts are represented with multiple rows. In XML subtags have no identifiers, while in a database each row is likely to have (an internal) one. Accordingly, a part would be represented with multiple rows sharing the same value of this internal identifier of the part.

location, etc. For the specific purpose of describing visual resources the Visual Resources Association VRA[10], a leading organization in the area of image management, has proposed a specialization of Dublin Core where nearly each Dublin Core property is elaborated into several properties specific for this domain. In eCulture we developed an RDF representation of the resources based on the VRA model.

We represent each work with an RDF object, an instance of VRA class vra:Work, and each image of the work with an instance of VRA class vra:Image, linked to the work with property vra:relation.depicts of the image. Accordingly, the example from Figure 1 should be represented in RDF as depicted in Figure 2.

```
<rdf:Description rdf:about="#SK-C-5">
 <rma:painter rdf:resource="#Rembrandt_Harmensz._van_Rijn"/>
 <dc:date>1642</dc:date>
 <dc:title xml:lang="nl">De Nachtwacht</dc:title>
 <dc:title xml:lang="nl">Het korporaalschap...</dc:title>
 <vra:title.alt xml:lang="en">The company ...</vra:title.alt>
 <rdf:type rdf:resource="vra#Work"/>
</rdf:Description>

<rdf:Description rdf:about="#id=M-SK-C-5-00">
 <vra:relation.depicts rdf:resource="#SK-C-5"/>
 <rdf:type rdf:resource="vra#Image"/>
</rdf:Description>

<rdf:Description rdf:about="#id=M-SK-C-5-01">
 <vra:relation.depicts rdf:resource="#SK-C-5"/>
 <rdf:type rdf:resource="vra#Image"/>
</rdf:Description>
```

Fig. 2. Sample RDF description of 'The Nightwatch' by Rembrandt

For vocabularies we use the Simple Knowledge Organization System (SKOS), a W3C model for expressing structure of thesauri and other vocabularies in RDF. It defines the major properties needed to represent vocabulary terms, such as skos:prefLabel (preferred label), skos:broader (link to a broader term), etc.

The following example illustrates how term 'Leiden', the birth town of Rembrandt, may be represented in SKOS:

```
<skos:Concept rdf:about="#Leiden">
  <skos:prefLabel xml:lang="nl">Leiden</skos:prefLabel>
  <skos:broader rdf:resource="#Netherlands"/>
</skos:Concept>
```

[10] http://www.vraweb.org/

4 Conversion

From Figure 1 and Figure 2 one can see a number of modeling differences:

- tags `record` are converted to RDF objects that are identified according to XML tag `object.number`,
- (potentially multiple) makers are converted to new objects and their roles are converted to RDF properties (`rma:painter`),
- (potentially multiple) reproductions (tag `reproduct`) are converted to new objects of type `vra:Image` and linked to works,
- etc.

These differences are quite common in practical conversion.

4.1 Object conversion

First of all, we need to find the source objects: the tag that wraps the objects up, and the tag that holds object identifiers (`record` and `object.number` in our example). In vocabularies the terms are often identified with their labels. We use an existing methodology [van Assem et al., 2004] to represent them as RDF objects.

To do this we developed a special conversion rule, the `ConvertObject`, that is responsible for creating target RDF objects and apply (property- and value-level) rules to flesh them up. In this rule we also select the objects to be converted:

- based on field values of the record in question;
- based on external lists of records that have online images.

It is common that some part of a dataset, up to as large as 80% may be filtered out because it is either not yet meant to be published, or not yet approved for publishing, or has too few annotations, or, finally, has no image of the cultural object being made and available online.

Source objects may contain repeating parts to be converted according to one of the following options:

- a part may need to be converted into a separate object linked to the main object. For example, tag `reproduct` is converted to an instance of class `vra:Image` linked to the whole with property `vra:relation.depicts`.
- a part may be converted to a specific RDF property. For example, tag `title` is converted to properties `dc:title` and `vra:title.alt`.
- a part may be converted to an RDF property, which name depends on another tag, such as the `maker` converted to property `rma:painter` according to tag `maker/maker.role/term`.

These operations are performed by the `ConvertObject` as well.

A number of other conversion rules were created and used to convert eCulture repositories. These rules and their usage statistics are presented in Table 1. There the rules are grouped into: vocabulary lookup rules, value translation rules,

sequential ordering of other rules, property renaming rules, branching of other rules, and custom rules.

Table 1 lists a number of datasets, where each dataset may consist of works meta-data, a local vocabulary of terms, and a local directory of artists. For each conversion rule the table shows the number of times this rule was used in each dataset-specific converter. These counts are summarized in three ways: the totals per rule and per datasets, the number of datasets depending on each rule, and the number different kinds of rules needed to convert a dataset that called 'diversity'.

		Datasets															
Rule groups	**Conversion rules**	aat terms	biblio terms	biblio works	geo terms	icn works	rkd artists	rma artists	rma terms	rma works	rmv works	svcn terms	tropen terms	tropen works	**Total**	**Dependent**
Vocabulary	LookupTerm		1	5	1	1	1		4	7	2	3	2		27	10
Value	AffixValue			1		1				1	2			2	7	5
Value	UseOtherPropertyValue											2	3		5	2
Value	ReplaceValues			1			1	1							3	3
Sequence	Sequence		1	4	1	1	4	2	2	4	1	2	2	1	25	13
Property	RenameLiteralProperty	3	9	13	3	4	14	9	4	29	7	1	1	10	107	13
Property	CreateResourceProperty	2	1	4	1	1	5	1	1	3	2	3	3	2	29	13
Property	RenameResourceProperty		3	1	4		2	1	1	4					16	7
Property	CreateLiteralProperty			1		1		1	1	2					6	6
Property	ExtractLiteralValueByPattern										1			1	2	2
Property	ExtractResourceValueByPatt											1	1		2	2
Branch	BranchOnPattern	1									2	6	5	2	16	5
Branch	FacetRenameProperty						1			1	4	1	1	3	11	7
Object	CreateNewObject			2			4								6	2
Custom	MakeEcultureImage			1		1				1	1			1	5	5
Custom	MakeEculturePerson						1	2		2			1		6	4
Vocabulary	LookupTermsFromText									2					2	1
Property	AAT$2	1													1	1
Branch	BranchOnTermInVocabulary												1		1	1
Custom	AAT$MakeAatSubject	3													3	1
Custom	RKDArtists$3						1								1	1
Custom	Biblio$ExtractRoleAndName			1											1	1
Object	RKDArtists$4						1								1	1
	Totals	10	15	34	10	10	35	17	13	56	22	19	20	22	283	
	Diversity	5	5	11	5	7	11	7	6	11	9	8	10	8		

Repositories: aat - AAT, biblio - Bibliopolis (`bibliopolis.nl`), geo - Geonames (`geonames.org`), icn - ICN (`icn.nl`), rma - Rijksmuseum, svcn - SVCN vocabulary (`svcn.nl`), tropen - Tropenmuseum (`tropenmuseum.nl`).

Table 1. The usage of conversion rules.

4.2 Object schema conversion

We will now briefly sketch the conversion rules that were reused.

Property rules. As can be seen from Table 1, property transformation rules (group `Property`) account for more than half of all rules written. Rule `RenameLiteralProperty` creates a literal RDF property and stores there the source value, leaving in intact. This rule is used most in the converters (107 times out of 283), and all the 13 datasets depend on it. Its sibling, rule `RenameResourceProperty` creates a resource RDF property, adding a namespace prefix to the value. It is used just 16 times.

The other rules in this group create literal and resource constants (rules `CreateLiteralProperty` and `CreateResourceProperty`), and extract values with a regular expression pattern (rules `ExtractLiteralValueByPattern` and `ExtractResourceValueByPattern`). They are used less frequently.

Rule ordering. Two types of ordering rules: sequential (rule `Sequence`) and branching (rules from group `Branch`) were used in the converters. The sequential rule simply executes other rules in a sequence, and the branching rules evaluate a condition to choose one rule to execute, out of a pool of options.

Rule `BranchOnPattern` evaluates if the value of a specified source property matches a regular expression pattern provided with the rule to choose one of two options to continue.

Rule `FacetRenameProperty` is designed for the frequent situation where the target property name should be chosen based on the value of some other property of the work. In our example the role of the maker (tag `maker.role/term`, value `schilder`, Dutch for 'painter') should be converted to property `painter`. Another role, represented with another value of tag `maker.role/term` would need to be converted to a different property.

Value replacement. Several value replacement rules proved their usefulness, as shown in rule group `Value`. These rules deal with prefixing and affixing values (rules `ValueDePrefixer` and `AffixValue`), perform a value replacing based on fixed table (rule `ReplaceValues`), and may take value of another property of the same object (rule `UseOtherPropertyValue`).

Custom code. There are several rules shown in Table 1 below the double line that are used in a single dataset each. These are specific rules that cannot be reused further.

4.3 Term alignment

Rule `LookupTerm` is used in most datasets. While it counts for just 10% of the total rule base, this rules generates all cross-collection links, providing all the bridges between collections. Let us sketch the basic principles that we use to do vocabulary lookup and assess its practical utility.

Term extraction. The terms are used as values of some properties, where they occur in two situations in the meta-data:

– the property may explicitly contain terms, where property name suggests the vocabulary they are coming from, e.g. `creator='Rembrandt'`;
– the property may be a title or description and have terms mentioned in it as text, e.g. `title='Portret of Rembrandt'`.

In the first case no natural language processing is needed, as typically each property value corresponds to a single term (multiple semicolon-separated terms was the most complex case here). However, they need to be cleaned up: trimmed, brackets removed, etc. It is also quite common to find some clarifications included in term labels in brackets. For example, paper in AAT is referred to as `paper (fiber product)` to differentiate it from other papers, such as the research paper you are reading now. We hard-coded the cleaning rules.

In the second case we extract words from the textual descriptions and try to match them to vocabulary terms.

Term alignment. Interpreting context of the terms is the key to our alignment strategy. Here we use the fact that in the cultural heritage domain the terms are defined with labels augmented with a context. This context consists of the following:

– term label and its language,
– property name that uses this term,
– other properties of the resource (work or thesaurus entry) using the term,
– analysis of the other terms used in this property,
– implicit context of the dataset.

We find a counterpart of a term we, first, need to select the vocabulary to lookup. It is typically known from the name of the property and a brief look at its values. Properties that refer to places are likely to occur in a geographical vocabulary (TGN or Geonames), people names in a directory (ULAN or RKD), and other terms, such as styles or materials – in a generic thesaurus like AAT. For example, property `material` is likely to correspond to the materials subtree of AAT, or `place.origin` to a geographic place. Implicit context of the dataset may allow refining that further, e.g. to restrict the search for the origin places to Egypt, in the case of an Egyptian collection.

Within the vocabulary we rely on exact string match of the labels to find possible counterparts. While typically regarded as too strict, it works in the cultural domain where vocabularies often provide many alternative labels for each term. For example, ULAN lists more than 40 ways of writing the name of Rembrandt, and it is quite likely that it will contain the name used by a specific museum.

An analysis of other terms helps in narrowing the part of a vocabulary that we need to search. For example, property `birth_place` with values like `Leiden` or `Amsterdam` suggests to store terms-cities. It should be aligned to cities from a geographical vocabulary TGN, while other geographic objects presented in the vocabulary, such as rivers, should be left out. In another example a property may

contain values such as `Frankrijk` that suggests that (i) these terms correspond to countries rather than cities, and (ii) their labels are given in Dutch.

The other properties of the resource using this term label are also quite useful. Let us look at a few typical cases:

- It is common to have two properties for places, one for country, and another for city. We first map the country, and then restrict the search for cities with the cities within this country. Here we exploit the part-of nesting typically provided by geographic vocabularies.
- For paintings we often know its creation date. Here we disambiguate possible painters by relating the creation date to their lifetime.
- When aligning directories of artists we compare not only their names, but also their lifetime years and even birth-death places for better disambiguation.

Using term context for disambiguating AAT terms seems to be more difficult than disambiguation of places and people.

4.4 Assessment

Schema conversion is easy to assess: it is either done or not. The number of properties is limited and can be assessed manually.

Our eCulture experience allows estimating the cost of conversion. It is realistic to estimate a single rule being written in an hour. To convert the Rijksmuseum repository of works, terms, and artists 86 rules are needed as shown in Table 1 (and other museums come at similar count). These 86 rules count to more than two weeks of working time, plus some time for the few custom rules. Accordingly, we can estimate one month of a conversion specialist being needed to convert an elaborated dataset.

Vocabulary alignment is more difficult because in each of its 30 usages the `LookupTerm` was applied to a specific property of a specific dataset that uses a specific part of a specific vocabulary. We do not make any statistical analysis of the mapping success as each of these 30 mapping cases is very specific, and any results aggregation would be as informative as the average patient temperature in a hospital. However, let us look at a few mapping cases presented in Table 2.

Table 2. Cases of thesauri alignment, '000

Repository - Vocabulary (concept count)	AAT (34)	TGN (1100)	ULAN (131)
Rijksmuseum	7 of 43	7.8 of 29 records	13 of 56
Ethnographic SVCN	4 of 5	3 of 5.5	
Artists list RKD			41 of 410
Bibliopolis	0.4 of 1.1		0.25 of 1

Rijksmuseum Amsterdam has developed an in-house thesaurus of 43.000 terms that we tried to map to AAT. At the moment, the mapping technique

described above gave us 7.000 mappings, as shown in Table 2. Note, that the Rijksmuseum thesaurus is larger that AAT term-wise. From a random manual check it seems that most of the terms that are not mapped simply do not occur in the AAT, or are phrased differently. Other vocabularies, such as SVCN fit AAT closer, and here we mapped 80% of the terms.

Aligning directories of people is also shown in Table 2, where 25% the Rijksmuseum people and 10% of the RKD[11] people were mapped to ULAN, that amounts to one third of ULAN records. In practical terms this is a good result as well.

Among the three kinds of vocabularies (terms, places, and people) places are the easiest as they have specific context and are organized. We map 3.000 out of 5.5000 places in SVCN to TGN. In addition, mapping TGN to a comparable Geonames vocabulary gives us, for example, 50% success in mapping Dutch places. Again, the real overlap is not known, so this success is difficult to judge.

Here we are talking about semantic mappings, e.g. based not only on exact match of the labels of the terms, but also on the interpretation of some contextual properties of these terms.

5 System

To implement the converters we developed `AnnoCultor`,[12] a Java-based system that provides the conversion infrastructure and implements a number of reusable conversion rules, shown in Table 1 and discussed in this paper. It is open for the inclusion of custom rules, modifying existing rules, and integrating with external systems, such as GATE.[13]. `AnnoCultor` is released as open source under GPL and is freely available for download.

To construct a converter for a specific dataset, one has to create a simple Java program, where the existing rules need to be put together. This does not require complicated Java programming: converters are created given a template by composing reusable rules.

6 Conclusions

The task of porting a cultural dataset to the Semantic Web consists of two subtasks: converting database schemas and aligning vocabulary data.

Programming required. As we practically proved in the eCulture project, the problem of integrating various database schemas in the cultural heritage area can be solved by converting the schemas to open web standards SKOS and specializations of Dublin Core. This conversion cannot de defined automatically. Moreover, the complexity of the conversion task requires writing conversion programs in a programming language.

[11] The national directory of Dutch artists, `rkd.nl`

[12] `http://annocultor.sourceforge.net`

[13] `http://www.gate.ac.uk/`

One man-month per dataset. Given our experience we can estimate around 4 weeks needed for a skillful professional to convert a major museum database (assuming that the `AnnoCultor` conversion system will be used). For this (s)he will have to create a converter of 50-100 rules plus some custom code.

Reasonable alignment success. Existing technologies, including the `AnnoCultor` tool discussed in this paper allow finding semantic alignments between works, vocabularies and artists. Success of the alignments depends on the specific vocabulary, collections and the terms that happen to be used, as these vary a lot. However, it is possible to achieve reasonable 50-80% of terms being mapped to the standard vocabularies without any advanced natural language processing, but with the use of the term context.

References

[Butler et al., 2004] Butler, M. H., Gilbert, J., Seaborne, A., and Smathers, K. (2004). Data conversion, extraction and record linkage using xml and rdf tools in project simile. Technical report, Digital Media Systems Lab, HP Labs Bristol.

[Kondylakis et al., 2006] Kondylakis, H., Doerr, M., and Plexousakis, D. (2006). Mapping language for information integration. Technical report, ICS-FORTH, `http://www.ics.forth.gr/isl/publications/paperlink/Mapping_TR385_December06.pdf`.

[Papotti and Torlone, 2004] Papotti, P. and Torlone, R. (2004). An approach to heterogeneous data translation based on xml conversion. In *Proceedings of the Int. Workshop on Web Information Systems Modeling at CaiSE-2004*, Riga.

[Schreiber et al., 2006] Schreiber, G., Amin, A., Assem, M., Boer, V., Hardman, L., Hildebrand, M., Hollink, L., Huang, Z., Kersen, J., Niet, M., Omelayenko, B., Ossenbruggen, J., Siebes, R., Takema, J., Tordai, A., Wielemaker, J., and Wielinga, B. (2006). Multimedian e-culture demonstrator. In *International Semantic Web Conference*, volume 4273 of *LNCS*, pages 951–958, Athens, GA.

[Sperberg-McQueen and Miller, 2004] Sperberg-McQueen, C. M. and Miller, E. (2004). On mapping from colloquial xml to rdf using xslt. In *Extreme Markup Languages 2004*.

[Tordai et al., 2007] Tordai, A., Omelayenko, B., and Schreiber, G. (2007). Thesaurus and metadata alignment for a semantic e-culture application. In *Proceedings of the 4th international conference on Knowledge capture (KCAP-2007)*, pages 199–200, Whistler, British Columbia, Canada.

[van Assem et al., 2004] van Assem, M., Menken, M. R., Schreiber, G., Wielemaker, J., and Wielinga, B. (2004). A Method for Converting Thesauri to RDF/OWL. In McIlraith, S. A., Plexousakis, D., and van Harmelen, F., editors, *Proceedings of the Third International Semantic Web Conference (ISWC'04)*, number 3298 in Lecture Notes in Computer Science, pages 17–31, Hiroshima, Japan. Springer-Verlag.

[Wielemaker et al., 2007] Wielemaker, J., Hildebrand, M., and van Ossenbruggen, J. (2007). Using Prolog as the fundament for applications on the semantic web. In S.Heymans, A. Polleres, E. R. D. P. and Gupta, G., editors, *Proceedings of the 2nd Workshop on Applicatiions of Logic Programming and to the web, Semantic Web and Semantic Web Services*, volume 287 of *CEUR Workshop Proceedings*, pages 84–98. CEUR-WS.org.